Limits and restrictions

This set of directives allow you to add restrictions to be applied when a client attempts to access a particular location or document on your server. Note that you will find additional directives for restricting access in the next chapter.

limit_except

Context: `location`

This directive allows you to prevent the use of all the HTTP methods except the ones that you explicitly allow. Within a `location` block, you may want to restrict the use of some HTTP methods, for instance by forbidding clients from sending POST requests. You need to define two elements—first, the methods that are not forbidden (the allowed methods; all others will be forbidden), and second, the audience that is affected by the restriction:

```
location /admin/ {
    limit_except GET {
      allow 192.168.1.0/24;
      deny all;
    }
}
```

The preceding example applies a restriction to the `/admin/` location—all visitors are only allowed to use the GET method. Visitors that have a local IP address, as specified with the `allow` directive (detailed in the HTTP Access module), are not affected by this restriction. If a visitor uses a forbidden method, Nginx will return a `403 Forbidden` HTTP error. Note that the GET method implies the HEAD method (if you allow GET, both GET and HEAD are allowed).

The syntax for this is as follows:

```
limit_except METHOD1 [METHOD2…] {
  allow | deny | auth_basic | auth_basic_user_file | proxy_pass |
perl;
}
```

The directives that you are allowed to insert within the block are documented in their respective module section in *Chapter 4, Module Configuration*.

limit_rate

Context: `http`, `server`, `location`, `if`

This directive allows you to limit the transfer rate of individual client connections. The rate is expressed in bytes per second:

```
limit_rate 500k;
```

This will limit the connection transfer rates to 500 kilobytes per second. If a client opens two connections, the client will be allowed 2 * 500 kilobytes.

Syntax: Size value

Default value: No limit

limit_rate_after

Context: `http`, `server`, `location`, `if`

This defines the amount of data transferred before the `limit_rate` directive takes effect.

```
limit_rate 10m;
```

Nginx sends the first `10` megabytes at the maximum speed. Past this size, the transfer rate is limited by the value specified with the `limit_rate` directive (see preceding section). Similarly to the `limit_rate` directive, this setting applies only to a single connection.

Syntax: Size value

Default: None

satisfy

Context: `location`

The `satisfy` directive defines whether clients require all access conditions to be valid (`satisfy all`) or at least one (`satisfy any`).

```
location /admin/ {
    allow 192.168.1.0/24;
    deny all;
    auth_basic "Authentication required";
    auth_basic_user_file conf/htpasswd;
}
```

In the preceding example, there are two conditions for clients to be able to access the resource:

- Through the `allow` and `deny` directives (HTTP Access module), we only allow clients that have a local IP address; all other clients are denied access.
- Through the `auth_basic` and `auth_basic_user_file` directives (HTTP Auth Basic module), we only allow clients that provide a valid username and password.

With `satisfy all`, the client must satisfy both the conditions in order to gain access to the resource. With `satisfy any`, if the client satisfies either condition, they are granted access.

Syntax: `satisfy any | all`

Default value: `all`

internal

Context: `location`

This directive specifies that the `location` block is internal. In other words, the specified resource cannot be accessed by external requests.

```
server {
    […]
    server_name .website.com;
    location /admin/ {
        internal;
    }
}
```

With the preceding configuration, clients will not be able to browse `http://website.com/admin/`. Such requests will be met with `404 Not Found` errors. The only way to access the resource is via internal redirects (check the *Rewrite module* section for more information on internal redirects).

File processing and caching

It's important for your websites to be built upon solid foundations. File access and caching is a critical aspect of web serving. In this instance, Nginx lets you perform precise tweaking with the use of the following directives.